January 1995

OS-9 International

Vol. 4, Issue 1

EFFO OS9

European Forum For OS-9

8606 Greifensee, Switzerland
Fax +41 1 940 38 90
email os9int@effo.ch

sFr. 10.00
ISSN: 1019-6714

startup

Recently, I tried to format a 2-GByte hard disk on an OS-9 system. Amongst others, the format program reported a disk size of -2,147,288,576 Byte. This somewhat obscure information could not be avoided; every time I called the format program, irrespective of the specified options, I was confronted with this negative number.

I then told this story to some of my friends at EFFO. They quickly put forward a possible explanation: "Microware has defined the variable that holds the disk size as signed long word instead of an unsigned one, that's obvious!"

Obvious!? Other operating systems simply define this variable correctly and allow formatting of hard disks with a capacity of up to 4 GByte. The people at Microware who designed the OS-9 operating system have solved a lot of more challenging problems – so why should they have overlooked this simple one?

I then studied all the available OS-9 manuals – no hints of negative sectors, but I always had the strange feeling that there is a deeper meaning in these negative numbers, and I created the following hypotheses:

- Microware intentionally limited the disk size to protect us against data losses: when a small disk crashes, less data is lost than when a big one crashes, is this not true?
- Microware realized the well-known proverb that says "The steady state of a disk is full".
- Microware anticipated the present environmental problems and protects the world from data garbage.
- Microware is planning to introduce a revolutionary post-modern computer system that for the first time integrates computer science **and** philosophy.

Old-fashioned computer systems cannot attribute a metaphysical value to electronic data. The availability of negative disk sectors would allow to store the data in relation to the value of any given dual system. Possible dual systems that may be used are

Good		Bad
	Matthew 26:41	
Heaven		Hell
Matter		Antimatter
Big endian		Little endian
0x0D		0x0A

In practice, data of one category would be written to positive sectors and data of the opposite category to negative ones. There is no ambiguity with sector zero, since this is needed for the boot information that would be located in the middle of the disk. A newly-developed transcendental random block file manager (TRBF) would take care of the assignment of data to one of the two disk areas. Fortunately, negative numbers have already been invented – so we look forward to this important OS-9 extension in version 4.0.

The hardware, of course, is eager, but the software is weak.

Hans-Werner Bippus

OmniRay

AUTOMATION MIT SYSTEM

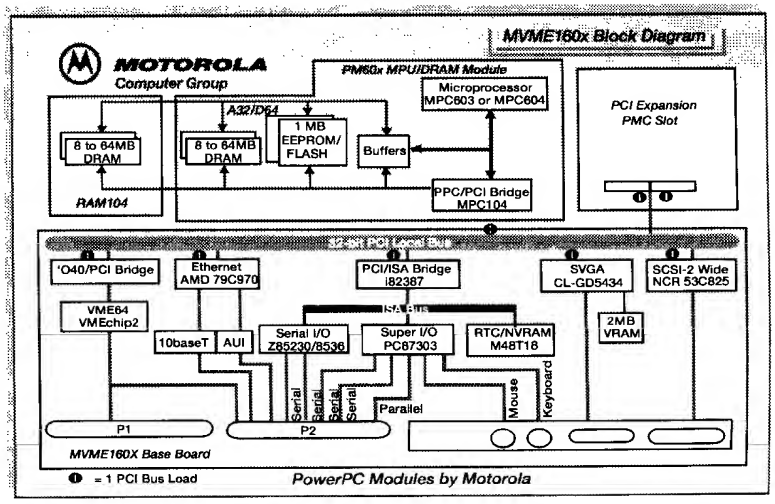
SY 01/95

More power for VME-boards thanks to the powerPC from Motorola.

The new product family MVME160x establishes new standards concerning price/performance as well as concerning maximum performance. Optimal flexibility is achieved by a modular design: processor-, memory- and mezzanine-modules for I/O are exchangeable.

The user has, for example, the choice between CPU-module with powerPC603 or -604 (66 resp. 100 MHz clock frequency) and 8 to 128 Mbyte DRAM that are mounted on the base board with the I/O controllers.

The base board offers maximum functionality with Ethernet, 16-bit SCSI-2, superVGA-graphics, mouse-port, keyboard-port and four serial-ports in one single VMEbus-slot. OS-9 is released on these boards.



Omni Ray AG

Industriestrasse 31, CH-8305 Dietlikon/Zurich, Phone 01 835 21 11, Fax 01 833 50 81

A Company of Sonepar Electronique International SEI

Peter Dibble at CERN

Douglas Kemp

Peter Dibble, computer scientist at Microware, visited CERN on Friday, November 18, 1994. He was accompanied by Nicholas Rainey, the General Manager of Microware Systems France. Peter Dibble's interesting talk reviewed Microware's recent releases and discussed plans for the near and far future.

Present and Near Future

Ultra C

There is a major release of Ultra C (1.2.1) which was unfortunately delayed to ensure compatibility with the Power PC (PPC) version. It has a much better assembler interface and improved code inlining (especially operating system calls). The floating point support package *fpsp* was optimized, the exception handling was tuned and the compiler has better long-word and cache-line alignment capabilities.

OS-9 Insights

The 3rd edition of Peter's book is now available. At least in the United States, there is an update offer, whereby one can send in the cover page taken from a previous edition and pay a reduced amount ($40) rather than the full price ($75). This book covers OS-9 version 3.0, as the edition number suggests.

Power PC

The OS-9 version for the Power PC 601 RISC processor was shown at the Open Bus Systems exhibition in Paris a few days after Peter Dibble's visit at CERN. While the original OS-9 release did not support SCSI and Ethernet, both features have been implemented as of end 1994. Programming for the PPC platform still requires one of Microware's cross development systems. Native development facilities and the source-level debugger are announced but not yet released. Versions for other members of the PPC family, such as the Power PC 603, are already running. A 25-MHz PC 601 benchmarked promising 109,000 Dhrystones.

OS-9 3.0.1 and ISP

The OS-9 3.0.1 maintenance release is announced, it includes support of Motorola's MC68360 micro controller as well as an improved version of the Internet Support Package ISP with several bug fixes.

FasTrak

The next release of this cross development package, due out in the first half of 1995, will include support for both MS Windows and Hewlett Packard's logic analyzers. Although these logic analyzers are very expensive, they are very useful for debugging and testing systems with minimal memory. It is also possible to debug a process that is already running – an often needed feature to find out why a task is looping etc. Future kernel releases will allow a debugger to attach to an already running process.

OS-9 Primer

Mark Heilpern, responsible for OS-9 training at Microware England for many years, has written the "OS-9 Primer" which not only addresses system programmers but also OS-9 users. Mark's book should be available in early 1995.

PC File Manager (*PCF*)

The next release of Peter's *PCF* will support the utilities *pd, dsave, free, deldir* and the SS_*Rename I$SetStt* call. There will be an option in the device descriptor allowing the driver to test various disk densities until it finds the correct one. Files can be written to sector zero and read back, hence user defined file systems can be installed or tar archive files can be generated for transfer to other systems. The problems of different end-of-line conventions being used by different systems can be handled by *readln/writeln* but not by *read/write*. The concept of "text mode" can be set for a file, afterwards it will be converted automatically. Because DOS lines are longer, they cause problems with concepts such as "length of write". This idea may be extended to NFS for OS-9 clients [1].

DAVID

The acronym DAVID stands for Digital Audio/Video Interactive Decoder. The set-top boxes supporting the digital audio/video interaction are a major development for Microware. A typical box handles 1.5 MBit/s MPEG encoded video channels distributed over phone lines or cable

networks. It contains a few megabytes of RAM along with the decoder and is expected to cost about $300 to $500. The devices already exist and have been demonstrated for about one year. Such equipment can easily be extended to other uses involving the transmission of data. Improvements in code size, performance and quality of OS-9 will all be very important in this market and, as such, will feed back into all versions of OS-9. Peter was happy to announce that the remote control of a DAVID box does not require the functionality of the Ctrl-Alt-Del key sequence.

Future

In the medium-term another PPC release is planned. The above mentioned support of "text mode" will be included in NFS and utilities.

The library support of *SS_Rename* will be added.

FasTrak will allow for system state source level debugging.

There will be a major restructuring of the boot ROM. In addition to boot modules found in ROM, NVRAM resident components will be detected and added to the boot menu. A small version of *IOMAN* can be expected, the tentative name *IOBOY* still awaits official confirmation.

Functionality and code of OS-9 and OS-9000 will merge gradually.

A new library containing most POSIX.1 calls except fork, followed by *POSIX* realtime extensions will improve the ability of users to write portable code.

A new C++ frontend to the Ultra C compiler is being investigated.

Ultra C 1.3 will include enhanced support for the MC68040 and MC68060 processors.

The debuggers will support a postmortem analysis of crashed processes.

Far Future

An assortment of asynchronous services will be added. For example, threads can currently not easily use the Random Block File manager *RBF*, unlike *SCF* with its support of signals.

Further significant improvements in kernel and compiler performance, support for more processor families and a native version of *FasTrak* can be expected.

References

[1] Emde C, Ocker W (1993) *Sharing Text Files via NFS.* OS-9 International 2(2):9-21.

Douglas Kemp is OS-9 coordinator at CERN. He can be reached by email at <kemp@dxcern.cern.ch>.

OS-9 Version 3.0 - What Is New?

Beat Forster

The current version 3.0 of the OS-9 operating system is being shipped since January 1994. In contrast to some other operating systems where improved versions are urgently needed and, therefore, rapidly installed, many OS-9 developers and most OS-9 users are normally not very eager to upgrade. One reason is that they do not suffer from important drawbacks so that they have time to await the experiences made by others. It is, therefore, the aim of this article to present the experiences made when porting OS-9 3.0 to a newly developed CPU board and using it over a time period of more than three months. The most important change between version 2.4 and 3.0 is Microware's license policy, but this is not the topic of this article.

Porting OS-9

There is one important change that must be applied to all drivers. Since OS-9 3.0 for the first time uses Motorola's master stack pointer (MSP) technique, the procedure to save the IRQ mask has changed.

The *init* entries of most driver sources use the following code section

```
move.b    M$IRQLvl(a1),d2      ; get IRQ level from descriptor
tst.w     d2                   ; does this device uses IRQ at all?
beq.s     NoIRQs               ; no
asl.w     #8,d2                ; yes, shift into priority
bset      #SupvrBit+8,d2       ; set system state bit
move.w    d2,IRQMask(a2)       ; save for future use
```

This code must now be replaced by

```
move.b    M$IRQLvl(a1),d2      ; get IRQ level from descriptor
tst.w     d2                   ; does this device uses IRQ at all?
beq.s     NoIRQs               ; no
asl.w     #8,d2                ; yes, shift into priority
move.w    sr,d0                ; get status register
andi.w    #%1111100011111111,d0; mask out current IRQ level
or.w      d0,d2                ; insert our IRQ level
move.w    d2,IRQMask(a2)       ; save for future use
```

Generally, there is no other modification required when old drivers are upgraded to OS-9 3.0. When file managers are upgraded, the newly introduced kernel pre-emption must be considered. The easiest solution is to disable pre-emption for the entire file manager, but it is also possible to initiate re-scheduling at safe places within the file manager.

Bus errors during module search in the boot procedure are no longer intercepted by the kernel as it was the case in OS-9 2.4. Such errors can occur when the boot file erroneously does not fit into the EPROM so that the calculated end of module would be located past the end of the addressable memory area. Under 3.0, this type of bus error can be prevented by using the newly available *padrom* utility prior to burning the EPROM.

Developing under OS-9

Most programs that are written under OS-9 2.4 will run without problems under OS-9 3.0, too. The following changes, however, must be considered.

- The file structure has changed drastically. The dream of many developers finally became reality, since there is now a clear-cut separation between host and target system. All system files needed to run the host system are properly separated from those necessary to compose bootfiles for target systems.

  The top level directory of this new structure is called *MWOS* (MicroWare Operating System). The lower levels are shown in the following tree:

```
MWOS
   OS9                    OS-9 specific files
      68000               processor type specific binaries
         CMDS
         LIB
         PORTS
      68020
         same as above
      etc.
         same as above
      SRC
         DEFS             OS-9 specific header files
   OS9000                 OS-9000 specific files
      etc.
   SRC
      DEFS                general header files
```

  All hardware manufacturers and other OS-9 distributors are encouraged to use and to support this directory structure.

- The field *P$PModul* (primary module) that can be examined using the *F$GPrDsc* system call contained the start address of all modules in version 2.4. From version 3.0 onwards, however, *P$PModul* of process number 1 (usually the kernel) points to the name entry of the module and no longer to its start address.

- The fields *P$MemImg* and *P$BlkSiz* of the process descriptor are no longer used. Instead, the newly available variable *P$Frags* contains the fragment list. If this list is going to be in-

spected in user mode and the *ssm* module is active, it must, of course, be made available through an *F$Permit* call.

- Structure and size of the device and IRQ table have changed. Programs that use either of these tables have to be recompiled. This is the reason why the *devs* and the *irqs* utilities from 2.4 do not run under 3.0 (see below).

- Microware no longer delivers the old C compiler *cc 3.2*; it is replaced by Ultra C. Unfortunately, new tools provided are not named differently although they are all but compatible to their precursors. This makes it impossible to develop simultaneously software for both 2.4 and 3.0 on the same system. In addition, recompilation of existing C programs requires that all makefiles be changed. Many libraries that have been produced with other compilers than those delivered by Microware are no longer compatible. Often used tools such as *liborder* and *libsplit* do not support 3.0 libraries. It may, therefore, take another 12 to 18 months until most of the OS-9 third party software is upgraded to OS-9 3.0.

- There is a problem with the *F$Sleep* kernel call. The following program

```
main()
{
  int i, retval;

  for (i = 1; i < 3; i++) {
    retval = tsleep(i);
    printf("tsleep(%d) returns %d.\n", i, retval);
  }
}
```

when running under 2.4 produces the correct output

```
tsleep(1) returns 0.
tsleep(2) returns 0.
```

Under 3.0, however, it returns

```
tsleep(1) returns 1.
tsleep(2) returns 0.
```

even if the *tsleep* function has not been awakened prematurely. Up to now, no other argument than 1 has been observed to cause this irregularity. According to Microware, the problem has been fixed in the drop-in version 3.0.1.

Using OS-9

No problems have been encountered when using the utilities from OS-9 3.0 on a 3.0 system and also on a 2.4 system. Most of the 2.4 utilities even run on a 3.0 system with the exceptions of the *procs*, *irqs* and *devs* utility. This is due to the above mentioned changes. The *procs* utility shows a *MemSiz* of zero for all processes, and the output from the *irqs* and *devs* utility is jammed.

By the way, the *setime* utility still has only a two-digit entry for the year: systems that are shipped today and aimed to survive longer than 5 years will need an update within the next five years. Otherwise it will not be possible to set the system's date after January 1, 2000. In case of EPROM resident systems, the *setime* utility may better be replaced by a more appropriate one.

The intercept handler no longer restores address register *a4* that the Omegasoft Pascal compiler uses as a global variable. All user supplied intercept routines must, therefore, restore *a4*. Apart from this change that does not interfere with usual programming strategies in Omegasoft Pascal, the compiler and all generated programs ran without problems. Unfortunately, maintenance of this fine compiler has been discontinued so that an official release for OS-9 3.0 will probably never be available.

Recommendations

- Before starting a new development project under OS-9 the responsible person should first check whether all required tools are available under OS-9 3.0. If so, it is a safe and powerful development platform that can be recommended without restrictions.
- If something essential is lacking, it might be better to start the development under OS-9 2.4 and to migrate later. Although an official upgrade strategy is not offered for the step from 2.4 to 3.0, special arrangements may be possible.
- In principle, the same as above applies to existing 2.4 projects. It should first be checked whether the project would profit from the enhancements available in OS-9 3.0. If so, advantage and disadvantage of the upgrade must carefully be evaluated. If not, it may be beneficial not to upgrade immediately.

Beat Forster works as a software and hardware engineer for a Swiss company. He can be reached by email at <beat@effo.ch>.

The GNU C Compilers, Part 3

Stephan Paschedag

Introduction

This is the last part in a series of articles about the GNU C Compiler (GCC) and the GNU C++ Compiler (GPP). It mainly describes the specific changes that have been applied to the compilers in order to cope with the differences between OS-9 and UNIX. In addition, enhancements have been introduced to facilitate the use of the compilers under OS-9.

The first article [1] summarized the history of the GNU C compilers and gave an overview about the compiler system in general, part 2 [2] described in detail the various stages of compilation, optimization and code generation.

Portability

The GNU compiler system is highly portable, since maximum attention was paid to separate all target specific code into a limited number of source files. GCC could, therefore, be ported to nearly all currently available processors and platforms. In addition, it is not only used as a host compiler but also as a cross compiler for other platforms or for embedded applications. Currently released versions even include support for digital signal processors and embedded controllers such as the Hitachi H8/300 family.

There is a total of five files, three header files, one source file and the machine description that contain system specific code. Two of the header files, *hconfig.h* and *tconfig.h*, contain among others *define* statements that describe the properties of the host and target systems. This distinction is necessary, since GCC can be configured to run on another system than the one its output is intended for (cross compiler). These settings include definitions such as

```
#define HOST_BITS_PER_INT 32
#define HOST_WORDS_BIG_ENDIAN
```

for the Motorola 68k family processors. The above defines that the processor uses 32-bit arithmetic and stores the most significant byte of a word at the lowest address.

Both the *tm.h* header file and the *aux-output.c* code file define properties of the target processor such as number of registers, register classes, recognition of addressing modes etc. The last file *md.h* contains the so-called machine description. This file is written in a special language that

allows to define patterns and constraints. They enable the compiler to match the internal RTL statements [2] with the target's instruction set. The following extract of the machine description gives the instruction pattern for testing a single integer *tstsi* of the MC68k processors.

```
(define_insn "tstsi"
  [(set (cc0)
        (match_operand:SI 0 "nonimmediate_operand" "rm"))]
  ""
  "*
{
  if (TARGET_68020 || ! ADDRESS_REG_P (operands[0]))
    return \"tst%.l %0\";
  /* On an address reg, cmpw may replace cmpl.  */
  return \"cmp%.w %#0,%0\";
}")
```

Every instruction pattern consists of four to five parts, a possibly empty name, an RTL template, a condition, the output template and an optional vector. In order to generate the compiler output in assembly language, GCC matches the individual RTL statements against the RTL template of the instruction patterns. Based on the operand information in the RTL template, GCC determines the type of parameters in the output template. In the above example a single integer operand *SI* is required that can be located either in a register or in memory *"rm"*.

The Port to OS-9

Normally, a specific configuration of GCC is selected using prepared installation scripts that create the above mentioned five files according to the desired target and host system. The make program is then called to build the compiler binaries. Unfortunately, OS-9 is not yet part of the official GNU distribution, since the currently available OS-9 implementation requires changes in files other than the ones intended for this purpose. In addition, the UNIX file name convention does not allow for an easy transfer of the source files to OS-9.

Unlike most other operating systems, OS-9 does distinguish between program and data space. Unfortunately, the original GCC does not support this distinction, since references to code and data are treated the same way. Various approaches have been used in the past to cope with this problem.

One approach that has been implemented in GCC V1.xx defines a second class of references so that there is one for code (SYMBOL_REF) and one for data (DATA_REF). Whenever GCC needs to create a reference it is assigned to one of the two classes. This class information is used later on during the final code generation phase to address a symbol either via program counter *pc* or via the global data pointer *a6*. The disadvantage of this procedure is that it requires changes to most of the GCC source files.

Another approach implemented in GCC V2.xx modifies the way references are generated. The following function call shows how the original GCC inserts a symbol reference into an RTL statement.

```
gen_rtx (SYMBOL_REF, Pmode, "label");
```

Under OS-9 this would result in an illegal reference to an absolute address. Therefore, the function call was changed into

```
gen_rtx (PLUS, Pmode, gen_rtx (SYMBOL_REF, Pmode, "label"), base_register);
```

where *base_register* is either the program counter *pc* or the data pointer. The simple reference is now replaced by the sum of the *base_register* and the symbol reference, which is exactly what OS-9 requires. The decision what to use as *base_register* is taken from the context. The advantage of this approach is that it required changes to only three of the source files. All other changes could be done in the previously mentioned five configuration files. In addition, it is now possible to optimize the entire construct which results in better code.

The above changes seemed to work perfectly until the first label had to be output in assembly language. The compiler generated the following line:

```
label(pc): move.l .....
```

Apparently, this was not the expected result. What had happened? GCC did not know better! It was only prepared for absolute addressing and, therefore, did not expect symbols to be defined as a sum. To correct this problem it was necessary to find all locations in the GCC-source where plain symbol names are output. Fortunately, there were only a few of them. After this step the hardest part was finished and a working compiler was available.

Extensions for OS-9

Experience gained from testing the above described compiler was promising but showed that it was not yet prepared for everyday's use. In detail, several OS-9 options and features such as the remote storage class and source level debugger support were considered to be absolutely necessary. While the remote storage class was relatively easy to implement and is now available through the *-mremote* command line option, source level debugger support was difficult to realize. In principle, the latter could be realized in two different ways. One of them was to port the GNU source level debugger *gdb*, the other was to teach GCC producing output that can be recognized by Microware's *srcdbg*. Although the internal structures of *srcdbg* are not officially documented, the second solution was preferred since rewriting the debugger interface was considered easier than porting *gdb*. In consequence, source level debugging is now available when the *-gg* or the *-gsrcdbg* command line option is specified.

Unfortunately, *srcdbg* is only prepared to accept source code in C language and not in C++. Using it in conjunction with C++ requires to realize that function names are mangled, i.e. the symbolic name is generated by appending the argument types to the original function name. This must be considered when setting breakpoints. For example,

```
int foo(int a, char ch) {

...
}
```

will appear as the label

```
foo__Fic: ...
```

in assembly language.

The availability of Ultra C and the fact that Microware has discontinued the release of *cc 3.2* libraries required further extensions to GCC. Unfortunately, Ultra C redefined the calling interface of C functions. While in *cc 3.2* the address registers *a0* and *a1* are generally available as scratch registers, i.e. a function does not need to preserve their contents, they are not in Ultra C. This restriction could lead to problems, if a function pointer was passed as argument across libraries made with different compilers. It was, therefore, a challenge to implement a calling interface compatible with the two Microware compilers. Hence, *gcc2* features now two additional command line options to control the calling interface.

-muccmode switch the calling interface to be compatible with Ultra C only, i.e. *a0* and *a1* are always preserved, and *gcc2* does expect them to remain unchanged throughout a function.

-muccmode2 switch the calling interface to be compatible with the two Microware compilers, i.e. *a0* and *a1* are always preserved, and *gcc2* does **not** expect them to remain unchanged throughout a function.

In consequence, if an Ultra C library is intended to be linked to a GCC compiled program, the *-muccmode* must be specified. If, however, a GCC compiled library is intended to be linked to a program compiled with either of Microware's compilers, the *-muccmode2* must be specified.

Enhancements for OS-9

Unlike UNIX where virtual memory is available, the stack size of an OS-9 application has to be defined at link time or when the application is started. Applications that do a lot of recursion or use large local data structures have to be given a large stack which may only be used under rare conditions. Therefore, *gcc2* implements a dynamic stack feature. If an application needs more stack than initially given, an additional stack segment is allocated. This segment is deallocated

automatically when the function returns. This feature allows for a more efficient use of the memory available in an OS-9 system. As an advantage, programs will no longer crash, if the required stack memory exceeds the amount estimated by the programmer. The penalty of this feature is a slightly higher execution time, but the implementation includes a cache-like memory allocation that prevents the stack memory from being allocated and deallocated too frequently. Dynamic stack sizing is enabled by specifying the *-mdynamic-stack* command line option.

Dynamic stack sizing works as follows: the stack memory of an application is located in the top area of its static storage. The application accesses its stack data using two pointers, the stack pointer and the frame pointer. The first one marks the top of the stack and is repositioned whenever a function allocates stack space. The frame pointer is normally used to access the local variables and the function arguments stored on the stack. The following example shows how this works. The source code segment in C language

```
void abc(int a, int b, char* p) {
    int x, y;
        def(x, a, p);
}
```

is compiled into

```
abc:    link      a5,#-8              ; create stack frame
        movem.l   a0/a1,-(a7)         ; save some registers
        move.l    d0,d1               ; copy a
        move.l    -4(a5),d0           ; get x
        move.l    24(a7),-(a7)        ; push p onto stack
        bsr       def                 ; call the function
        movem.l   (a7)+,a0/a1
        unlk      a5                  ; free stack frame
        rts                           ; return from function
```

which results in the stack frame:

```
        argument p
        return addr
a5->    old frame-pointer
        x
        y
        register a1
a7->    register a0
```

Since both the function's arguments as well as local variables are accessed using the same pointer, the stack can only be expanded contiguously. If the stack memory is exhausted, a new noncontiguous segment must be allocated so that local variables are now separated from the arguments. This requires the introduction of a third pointer, the so called argument pointer. The above C code than becomes

```
abc:    link      a4,#0              ; create argument frame
        link      a5,#-8             ; create stack frame
        movem.l   a0/a1,-(a7)        ; save some registers
        move.l    d0,d1              ; copy a
        move.l    -4(a5),d0          ; get x
        move.l    8(a4),-(a7)        ; push p onto stack
        bsr       def                ; call the function
        movem.l   (a7)+,a0/a1
        unlk      a5                 ; free stack frame
        unlk      a4                 ; free argument frame
        rts                          ; return from function
```

which results in the stack frame:

```
        argument p
        return addr
a4->    old arg-pointer
a5->    old frame-pointer
        x
        y
        register a1
a7->    register a0
```

The arguments are now accessed through the argument pointer *a4* and the local variables through the frame pointer *a5*. The stack can now be expanded and the argument pointer points into the old stack frame and the frame pointer points to the new one. The first few bytes of the new block are occupied by information to restore the previous frame at the end of the function

```
        argument p
        return addr
a4->    old arg-pointer
```

and somewhere else in memory

```
        pointer to old stack
        old stack-check info
        size of new stack segment
        return address to cleanup stack
        dummy arg-pointer
a5->    old frame-pointer
        x
        y
        register a1
a7->    register a0
```

As already mentioned [2], *gcc2* was enhanced with built-in rules to locate and name library files. To add a new path to the default library search list, either the environment variable *GCCLIB* can be set or the command line option *-L<path>* be specified.

Troubleshooting

The *gcc2* front end for OS-9 offers a variety of implicit rules and default settings for the inclusion and use of auxiliary files. The advantage of this policy is that the *makefile* can remain straightforward and easy to understand. If, however, the *makefile* does not execute correctly, the reason for this misbehaviour may not be obvious. The *gcc2* front end, therefore, offers several trace options to locate the source of the problem.

In detail the following options are available:

-v This switch makes *gcc2* verbose. The command lines of all compilation phases are displayed. In addition, the search list of the include files is listed in the same order as used by the preprocessor.

-v -Q These switches make *gcc2* even more verbose, as *-Q* disables the quiet flag that is enabled by default. In consequence, the compiler displays all built-in and specified flags and also the active machine-specific options. Furthermore, a protocol is generated presenting every function's name along with its compilation time.

-dm This switch causes the compiler to display the amount of memory used which can be useful on systems with a small amount of memory.

-H This switch forces the preprocessor to display all included files and their include level. A preceding dot indicates the level of inclusion and should not be mistaken as a relative directory path.

-debug This switch causes *gcc2* to show where it searches for the specified libraries and where they are found.

-E -Fcccp2o [-dD] This switch combination disables all compilation stages except preprocessing. In addition, macros and definitions are displayed.

Maintenance

Not only GCC but also the port to OS-9 is under continuous development. Since the quality of GCC for OS-9 mainly depends on the feedback from C/C++ programmers, remarks and error reports are needed for further improvement. Please send them by email to <stp@effo.ch> or by fax to EFFO.

References

[1] Paschedag S (1994) *The GNU C Compilers.* OS-9 International 3(2):13-25.

[2] Paschedag S (1994) *The GNU C Compilers, Part 2.* OS-9 International 3(3):13-22.

Stephan Paschedag works as a hardware and software engineer for a Swiss company. He can be reached by email at <stp@effo.ch>.

Big Hard Disks Under OS-9

Carsten Emde

When the OS-9 operating system was designed about 17 years ago, the standard disk medium was an 8" floppy disk that had a storage capacity of 638 kByte in the standard density mode. Using a sector size of 256 Byte, this resulted in a total of 2554 sectors on disk which is 0x09fa in hexadecimal notation. Other operating systems would have reserved exactly 12 bits to store this number on the disk root structure, but the people at Microware were optimistic enough to believe in the longevity of their product and to reserve twice as much. The length of this 24-Byte entry, in fact, now limits the maximum capacity of an OS-9 disk volume, and this appears easy to be calculated: 0xffffff (16,777,215) times the most common sector size of 512 Byte equals 8,589,934,080 Byte or, roughly 8.5 GByte.

Unfortunately, the correct value is much less; this article explains why.

The OS-9 random block file manager *RBF* that manages the random accesses to a disk volume needs to seek to specific disk locations. The related command *I$Seek* expects the seek offset as a 32-bit address of the next disk byte to be read or written. The OS-9 Technical Manual explicitly states that *RBF* would take a negative address as a large positive number, i.e. that in C notation it refers to unsigned long. This limitation makes it impossible to format a disk volume that contains more than 0xffffffff or 4,294,967,295 Byte.

Having this consideration in mind, it was tried to format a Seagate ST12400N hard disk with a nominal storage capacity of 2,160,000,000 Byte under OS-9. When the *format* command displayed the format data on screen (using the SCSI auto format feature) the surprised reader was confronted with the information that instead of positive sectors this disk uses negative ones:

```
———————  Format Data  ————————

Fixed values:
                Disk type: hard
              Sector size: 512
            Disk capacity: 4194685 sectors
                           (-2147288576 bytes)
            Sector offset: 0
             Track offset: 0
               LSN offset: $000000
 Minimum sect allocation: 32
```

In the hope that this is only a problem of the *format* program that erroneously specifies *%d* instead of *%u* in the *printf* control string, the format procedure was not aborted but continued. Unfortunately, the disk could not be formatted correctly. It was, therefore, concluded that the upper limit for the number of bytes on an OS-9 disk volume is not 0xffffffff but 0x7fffffff or

2,147,483,648 Byte. Since this is only marginally less than the expected hard disk capacity, it was decided to disable the auto format mode and to explicitly specify disk geometry data resulting in a total number of sectors that is only slightly lower than the above given maximum.

This time, the *format* program exited without error but, unfortunately, when more than about 16 MByte of data were written to disk, *RBF* (edition 97) stopped writing and returned error number 248. The utilities *free* and *dcheck*, however, still reported a nearly empty disk. This error could be avoided when the cluster size was set to 16 instead of the required 8 sectors per cluster. However, this method cannot be recommended generally, since the minimum file size would be 8 kByte and, therefore, much disk space would be wasted, if there is a big number of small files (less than the cluster size).

Recently, Microware has shipped a drop-in version of the *RBF* manager (edition 98). This version has the above problem fixed.

Recommendations

1. Hard disks with a capacity of up to about 1 GByte and 512 Byte per sector can be used without any problem. The required cluster size of 4 sectors ensures efficient use of the disk space.

2. If hard disks with a capacity of more than 1 GByte and less than 2 GByte are used, the *RBF* edition 98 must be installed. This *RBF* version runs under both OS-9 2.4 and 3.0.

3. If hard disks with a capacity of more than about 2 GByte are used, they must be partitioned into sections of less than 2 GByte. The sector offset entry (32-bit word at position 0x68 in the descriptor) allows for this feature. This is best edited in the descriptor using the *moded* utility.

Carsten Emde can be reached by email at <carsten@effo.ch>.

Remote Commands and Network Time Services for OS-9

Carsten Emde

Introduction

In addition to basic services provided by TCP/IP software packages such as *telnet* and *ftp*, other protocols and services have been developed to make a network connection as versatile as possible. Among others, these protocols include remote printer and remote streamer support, remote login, remote shell and network copy commands. Protocols such as SMTP (Simple Mail Transport Protocol) and SNMP (Simple Network Management Protocol) even allow sending electronic mail over the network and provide diagnostics and troubleshooting. Last but not least, programs have been developed that allow to synchronize the realtime clocks of different computers via network. Unfortunately, software to use these additional services is neither included in Microware's Internet Support Package nor is it commercially available elsewhere.

Remote printer support, remote shell and the network copy command are probably the most useful remote network commands. In addition, network time commands are also frequently needed. It is difficult to understand why these tools are not available under OS-9, while they can be found even on rudimentary Personal Computers. EFFO has, therefore, decided to create a new public domain disk that contains these tools for OS-9. The current article describes their principles of operation, installation on an OS-9 system and use in a heterogeneous network with OS-9 and UNIX computers.

Copyright

Implementation under OS-9

A special network library was used to implement the software under OS-9. Many ideas have been taken from the literature [1–3], but not all of them could be realized under OS-9. The most important difference between a standard UNIX implementation and the related OS-9 version is the way server programs write progress messages or error reports. In contrast to client programs, servers are either implicitly started from the *inetd* super demon or from the *startup* file or a similar procedure at boot time. In the latter case, they often run in the background with input and output path redirected to */nil*, for example

```
progd <>>>/nil&
```

In both cases, there is no standard definition where to write messages and error reports. Some OS-9 network packages provide the UNIX *syslog* functionality, but this may not be available on a particular system. It was, therefore, decided to use an approach that probably works on most systems: the server program first tries to inspect the environment variable *LOGDIR*; if set, the log file is opened on that directory under the name of the server program suffixed by *.log*. If *LOGDIR* is not set, the directory prefix */dd/LOG* is tried and, if also not available, */dd* is used for this purpose. Hence, on an EPROM-resident system that has no RAM disk installed, there is no message and error output, but these systems normally require client and not server functionality.

Client programs do not differ with respect to message and error output, they simply write to the standard error path as usual under both UNIX and OS-9.

Using Remote Commands

Remote Printer Support (*lpr*)

The *lpr* program requires an appropriate printer demon program *lpd* to be installed anywhere in the network. To initiate a printing procedure, *lpr* needs to know the host where the printer is installed, the name of the printer service on that host and the name of the file to be printed. The first two items can be specified in the command line using the *-h* and *-p* option, respectively, but they can also be specified in the environment using the *LPHOST* and *LPDEST* variables. The file name must always be specified as command line argument. To print the file *listing* using printer service *ps* on host *printhost*, for example, the following command line can be entered

```
lpr listing -p=ps -h=printhost
```

A better approach, however, would be to add the two lines

```
setenv LPHOST printhost
setenv LPDEST ps
```

to the user's *.login* file so that the above print command is reduced to

```
lpr listing
```

Remote Shell (*rsh* Client, *rshd* and *rshdc* Servers) and Remote Copy (*rcp*)

These remote programs can be regarded as a step between procedural file access via ftp and transparent file sharing via NFS. Procedural file access requires that a special file transfer program is entered and actions on remote files are only possible through this program. To inspect the directory of the remote host *sourcehost*, it is, for example, necessary to enter

```
$ ftp sourcehost
Connected to sourcehost.
sourcehost TCP server (Version 1.0 - 01.01.95) ready.
Name (host:name): name
Password required for name.
Password: ****
name login ok.
ftp> dir
PORT command successful.
Opening data connection for dir -ea (100.100.100.100,1000)

Directory of "."

etc.

Transfer complete.
4000 Bytes received in 2 seconds (2 Kbyte/s)
ftp>quit
$
```

Using NFS, the above procedure is much simpler; it is possible to directly use the local *dir* program and to enter

```
$ dir /sourcehost -ae
```

to obtain the same information as above. As a disadvantage, NFS requires a separate license and may need CPU upgrade and installation of additional memory.

A remote shell session is less complicated than ftp but less elegant than NFS. It allows to remotely start a shell that forks a program on the remote host having its output redirected to the local network path. To inspect the root directory of the remote host *sourcehost*, as in the above example, the appropriate *rsh* command would be

```
$ rsh sourcehost 'dir -ea'
```

or, if the remote computer is a UNIX station,

```
$ rsh sourcehost 'ls -al'
```

The above *dir* and *ls* programs are used as examples; any other non-interactive program such as *procs*, *mfree* etc. can also be used without restriction.

In principle, the remote copy command also represents a step between ftp and NFS. It is similar to the local copy program but offers the possibility to prefix the file name with the name of the host where the file is requested from. Host and file names are separated by a colon. Hence, the command line

```
$ rcp sourcehost:/dd/SRC/prog.c
```

would copy the file */dd/SRC/prog.c* from host *sourcehost* to the current working directory. File attributes and date stamps are preserved when the *-p* option is specified as in

```
rcp -p sourcehost:/h0/startup /r0/startup
```

Even entire directory structures can be copied recursively by specifying the *-r* option. All header files from the remote system *sourcehost* will, for example, be copied to the local */dd/USR* directory when the command

```
$ rcp -rp sourcehost:/usr/lib/local /dd/USR
```

is submitted.

Remote Daytime and Clock Synchronization via Network

The remote daytime service is called *remotime*; it simply displays the current daytime of the specified host.

To synchronize the realtime clock of the local computer with the time of a network time server, the *netime* command is available. In principle, it works like the *setime* utility except that it takes the time input from the specified host and not from command line. A special test option *-t* allows to only display the time that would be set.

Installing Remote Commands

The software may be copied from floppy disk to the system's hard disk without modifying the directory structure. The following files are available:

File	Function
/dd/.rhosts	user-specified file of equivalent hosts and users
/dd/CMDS/daytimed	remote daytime (server)
/dd/CMDS/lpr	remote printing (client)
/dd/CMDS/netime	remote time (client)
/dd/CMDS/rcp	remote copy service (server and client)
/dd/CMDS/remotime	remote daytime (client)
/dd/CMDS/rsh	remote shell (client)
/dd/CMDS/rshd	remote shell (server)
/dd/CMDS/rshdc	needed by the remote shell server
/dd/CMDS/timed	remote time (server)
/dd/SYS/hosts.equiv	names of hosts with "equivalent" user IDs

If some of the files, e.g. *.rhosts* or */SYS/hosts.equiv*, are already available, they should be renamed or copied elsewhere before the software is copied. The next two paragraphs describe how these files are adapted to meet the requirements of a particular system.

The File */dd/SYS/hosts.equiv*

The *rsh* server program *rshd*, used to support *rsh* and *rcp* requests, uses the file */dd/SYS/hosts.equiv* in the following way. When the connection is made, *rshd* gets the user ID of the user on the remote ("calling") system. It then looks up the remote user in the local */dd/SYS/password* file. If the remote user is not the super user, then */dd/SYS/hosts.equiv* is checked for the name of the remote host. If it is found, the user is considered to be equivalent to the user of the same local name, and the command proceeds. If the host name is not found, or if the remote user is the super user, then *rshd* checks the file *.rhosts* in the user's login directory found in */dd/SYS/password*. If an entry is found for the remote host, or for this local user name and remote host combination, then the user is considered equivalent and the command proceeds. If this test fails, the command is terminated and the error message "permission denied" is written to the client's standard error output path.

The format of */dd/SYS/hosts.equiv* is a list of host names, one per line. The "primary" host name, i.e. the first one in the *hosts* file, must be specified in this list.

The File *.rhosts*

This file resides in a user's login directory. It contains entries, one per line, which are of the form:

```
hostname username
```

It allows a user to specify a set of users on other systems which are allowed equivalent capabilities to himself or herself on this system. The username field is optional.

In an environment where a single organization might have various systems used by a common set of users, it is often the case that a single user wants to have login IDs on many different systems. In the common case where the login IDs are the same for each user on all systems, then user authentication is provided by the list of host names in the file */dd/SYS/hosts.equiv*. In the case where a host is not in the */dd/SYS/hosts.equiv* list, or the user has a different name on another system, the user can provide individual authentication by adding entries in his or her personal *.rhosts* file. Users who connect to the system via *rcp* or *rsh* and are authorized via *.rhosts*, will have privileges on this system exactly equivalent to the user granting authorization.

The *rshd* server program, used to support *rsh* and *rcp* requests, uses *.rhosts* in the following way. When the connection is made, *rshd* gets the user ID of the user on the remote ("calling") system. It then looks up the remote user in the local */dd/SYS/password* file. If the remote user is not the super user, then */dd/SYS/hosts.equiv* is checked for the name of the remote host. If it is found, the user is considered to be equivalent to the user of the same local name, and the command proceeds. If the host name is not found, or if the remote user is the super user, then *rshd* checks the file *.rhosts* in the user's login directory found in */dd/SYS/password*. If an entry is found for the remote host, or for this local user name and remote host combination, then the user is considered equivalent and the command proceeds. If this test fails, the command is terminated and the error message "permission denied" is written to the client's standard error output path.

The host name entry specified in *.rhosts* must be the "primary" host name, i.e. the first one in the *hosts* file.

Starting Remote Servers on an OS-9 System

Some OS-9 network packages provide a UNIX compatible *inetd* server that allows for providing network services just by enabling them in the *inetd.conf* configuration file. Details for this procedure are given in the next paragraph. If such an *inetd* super demon is not available, the *rshd* server is started simply by typing

```
$ rshd <>>>/nil&
```

or, better, by integrating this command into the system's startup procedure after the network system has been launched. A short protocol is written to the trace output *rshd.log* in the above mentioned directory. A longer trace output can be obtained, if the *-d* option is specified – the more *ds*, the longer the output. In addition to starting the *rshd* program, the *rcp* client and server program should be loaded into memory

```
$ load rcp
```

so that it can also be found when running under a user's login that does not specify the appropriate execution directory. Again, this command is best placed into the *startup* file.

Daytime and time servers are started similarly, i.e. the following two lines

```
daytimed <>>>/nil&
timed <>>>/nil&
```

are best placed into the *startup* file of the OS-9 system.

Starting Remote Servers on a UNIX System

On some OS-9 and on standard UNIX systems, the *inetd* super demon takes care of starting the appropriate servers. They are defined in the *inetd.conf* file. If a service is available by the *inetd* server itself, it is defined as internal; the fully qualified path name of a server program must be specified, otherwise. To enable the internal daytime service via the TCP protocol, for example, the line

```
time stream tcp nowait root internal
```

must be added to the *inetd.conf* file. Remote shell service using the *rshd* server in the */dd/ISP/CMDS/rshd* directory would, for example, require the line

```
shell stream tcp nowait root /dd/ISP/CMDS/rshd
```

in the configuration file.

Special Considerations for OS-9

Output from a UNIX program differs from an OS-9 program insofar as the line delimiter is not Carriage Return but Line Feed. This cannot be corrected by the remote command utilities, since the output of a local program is directly sent to the network port. As a cure in such cases, it is recommended to pipe the program's output through the *tr* utility, or to use the *tr* program instead of the *list* utility. If, for example, the content of an OS-9 *startup* file must be inspected from a UNIX host, the following command is appropriate

```
% rsh os9host 'tr \\n \\l /dd/startup'
```

In another example, the active OS-9 processes are displayed on a UNIX host using the command

```
% rsh os9host 'procs -e ! tr \\n \\l'
```

but it would also be possible to use the UNIX *tr* program for this purpose

```
% rsh os9host 'procs -e' | tr \\015 \\012
```

so that conversion from Carriage Return (hexadecimal 0x0d, octal 015) to Line Feed (hexadecimal 0x0a, octal 012) is not done on the OS-9 but on the UNIX system.

References

[1] Comer DE (1991) *Internetworking with TCP/IP Vol. I: Principles, protocols, and architecture.* Prentice-Hall, Englewood Cliffs.

[2] Comer DE, Stevens DL (1991) *Internetworking with TCP/IP Vol. II: Design, implementation, and internals.* Prentice-Hall, Englewood Cliffs.

[3] Comer DE, Stevens DL (1993) *Internetworking with TCP/IP Vol. III: Client server programming and applications.* Prentice-Hall, Englewood Cliffs.

Carsten Emde can be reached by email at <carsten@effo.ch>.

Non-blocking Write to a Pipe

Carsten Emde

Introduction

Pipes represent one of the standard OS-9 methods to exchange messages between two or more different processes. They are, for example, implicitly used, when the pipe symbol (!) is entered in a shell session. The shell input line

```
$ echo AAABBB ! tr A Z
ZZZBBB
```

is mostly equivalent to

```
$ echo AAABBB >/pipe/pipe_file & tr A Z </pipe/pipe_file
ZZZBBB
```

except that a named pipe is used in the second shell command whereas the shell creates an unnamed pipe in the first one.

One very important feature of pipes is that they can be used to synchronize the processes involved. In principle, this works similarly to two computers that are linked via a serial connection with a handshake protocol: the writing process can never write faster than the reading process can read and vice versa. The length of the input and output buffers defines how tightly the two processes are coupled. If pipes are used, there is only one buffer involved; it has a default size of 90 Bytes. In contrast to two serially linked computers where the communication is the main purpose of the connection, processes that are linked via a pipe have other main purposes such as plant control etc. They only use the link for synchronisation. It is, therefore, often mandatory that the writing process does not block, if an attempt is made to write into a pipe having not enough buffer space available to receive the requested amount of data. This is called non-blocking write. The following article explains how non-blocking write can be achieved when dealing with pipes.

Basic Principle

The principle is based on the dualism of the pipes' nature. On the one hand, the *pipeman* file manager behaves like the random block file manager (*RBF*), since pipe files can be queried using the dir command:

```
$ dir /pipe/pipe_file
Directory of "/pipe" 12:00:00  01/01/95

Owner    Last modified  Attributes Sector Bytecount Name
-------  -------------  ---------- ------ --------- ----
  0.0    95/01/01 1200   ------wr    EFFO         7 pipe_file
```

This is only possible, since *pipeman* appropriately fills the file descriptor structure that is returned by a *_gs_gfd* call. On the other hand, the pipeman behaves like the sequential character file manager (*SCF*), since data can be read and written only sequentially. The *SCF*-like behaviour is documented further by the fact that the number of bytes in a pipe file can be queried not only using the above named *_gs_gfd* call but also using *_gs_rdy*. The latter always returns -1 when passing the path number of an *RBF* device. But the use of the *_gs_rdy* call is also different between *SCF* and pipe devices. On an *SCF* device, *_gs_rdy* only tests the receive buffer and is never used in conjunction with the *write* command; on a pipe device, however, *_gs_rdy* can be used equally well with *read* and *write* calls: prior to a *read* call, *_gs_rdy* can determine the maximum number of bytes to be read, prior to a *write* call, the difference between the pipe buffer size and the return value of *_gs_rdy* determines the maximum number of bytes that can be written.

The functions that determine and set the size of an *RBF* file, *_ss_size* and *_gs_size*, are also different, if used in conjunction with pipes: the *_gs_size* function returns the size of an *RBF* file; if, however, the path number of a pipe device is passed, *_gs_size* returns the buffer size of that particular pipe and does not at all reflect the number of characters that are currently available. The *_ss_size* function sets the size of an *RBF* file; when dealing with pipes, only 0 is allowed which resets a pipe path provided the pipe has no active readers or writers. Any other size value is ignored.

There are two different ways to realize non-blocking writes into a pipe: one method is based on creating a pipe of known size [1], the other uses the *_gs_size* command. The two methods are exemplified in the following source code sections.

Pipes of Known Size

A pipe of known size can be created using the create function call.

```
#include <modes.h>

#define MODE S_IREAD|S_IWRITE|S_ISIZE
#define PERM S_IREAD|S_IWRITE
#define SIZE 192

extern int errno;

main()
{
  char *outstring = "I am a test string";
  char *pipename  = "/pipe/test";
  int   pipepath;

  if ((pipepath = create(pipename, MODE, PERM, SIZE)) < 0)
    exit(_errmsg(errno, "can't create pipe '%s' due to ", pipename));

  /* Let another process read from our pipe */
  /*...*/

  /* would the next write block the program? */
  while (strlen(outstring) > SIZE - _gs_rdy(pipepath)) {
    /* yes, do something else */
  }
  /* no, write is safe */
  write(pipepath, outstring, strlen(outstring));
}
```

The above example program only works, however, if the string *outstring* to be written is not larger than the pipe buffer *SIZE*. It must be noted that the actual buffer size is not always the same as specified in the *SIZE* argument; it cannot be less than 90 Bytes and will be rounded to the next 16-Byte boundary, if larger. It is also important to note that the conditional of the above *while* statement may not always be correct, since *_gs_rdy* returns -1 and not 0, if the pipe is empty. In addition, *errno* is then set to *E_NOTRDY* (246).

Pipes of Unknown Size

The above method does not work, if the pipe size is not known, e.g. because the pipe has been created by another process. For such cases, the *_gs_size* function is available so that the above code segment reads:

```
#include <modes.h>

#define PERM S_IREAD|S_IWRITE

extern int errno;

main()
{
  char *outstring = "I am a test string";
  char *pipename  = "/pipe/test";
  int   pipepath, maxpipesize;

  /* the pipe has been created by another process */
  if ((pipepath = open(pipename, PERM)) < 0)
    exit(_errmsg(errno, "can't open pipe '%s' due to ", pipename));
  maxpipesize = _gs_size(pipepath);

  /* let another process read from our pipe */
  /*...*/

  /* would the next write block the program? */
  while (strlen(outstring) > maxpipesize - _gs_rdy(pipepath)) {
    /* yes, do something else */
  }
  /* no, write is safe */
  write(pipepath, outstring, strlen(outstring));
}
```

Restriction

The proposed code segments have one restriction in common: they only work reliably, if there is no other process writing to the same pipe. Otherwise, it can never be excluded that the other process is writing to the pipe during the small time interval between the *_gs_rdy* and the *write* call. In such cases, it may become necessary to install events or semaphore events that lock all other *write* accesses to the pipe between the *_gs_rdy* and the *write* call. Another theoretical and probably more elegant solution to this problem would be to open the pipe path for non-sharable access and to close it again after writing. Unfortunately, *pipeman* currently does not support the *I_SHARE* permission bit.

References

[1] Dayan PS (1992) *The OS-9 Guru, 1 – The Facts*, edition 1, Galactic Industrial Ltd., Durham UK.

Carsten Emde can be reached by email at <carsten@effo.ch>.

The EFFO 1995 AGM

Reto Peter, EFFO Secretary

The 8th EFFO annual general meeting (AGM) took place Saturday, January 21, 1995, 14:30 - 20:30 at the Gasthof zur Herberge in Teufenthal (near Aarau). All registered EFFO members had received written invitations including the proposed agenda and the proposed budget for 1995.

The president Werner Stehling welcomed all present members. First of all, he expressed his gratitude to all active EFFO members who wrote articles for OS-9 International, produced software and documentation or provided other services. He then gave a short summary of EFFO's activities in 1994. The number of members has slightly increased to about 70; the shift from private OS-9 users towards professional developers has continued.

The support EFFO is getting from companies is also increasing. Apart from Eltec and Elsoft who continued their already given support, other companies such as PEP and Microware France recently discovered EFFO.

There is a steady demand for PD disks; most orders ask for more than one disk, some of them even request the complete disk set.

EFFO took over the responsibility for the journal OS-9 International. Producing and publishing 3 issues in 1994 as promised consumed most of the time active members invested in EFFO.

Formal Points of the Agenda

The profit and loss account and the balance for the last year were presented, verified by the auditor and accepted unanimously.

EFFO's responsibility for OS-9 International required changes to be made to some of the articles. These changes have been accepted with one vote.

The elections of the officers did not require extensive discussion, since all nominees were elected with the maximum of votes possible. For the first time and as a consequence of the changes in the EFFO articles, an officer had to be elected as Editor-in-Chief of OS-9 International. The members of the committee for 1995 are:

President	Werner Stehling (as before)
Vice-president	Reto Peter (as before)
Secretary / registrar	Reto Peter (as before)
Treasurer	Stephan Paschedag (as before)
Editor-in-Chief	Carsten Emde (new)
Auditor	Hans-Werner Bippus (new)

The prices of PD disks needed to be adjusted. Their prices increased to sFr 15.- (sFr 10.- for EFFO members) per disk. These prices were approved together with the unchanged prices for OS-9 International and the annual subscriptions for EFFO membership. Also in 1995, members from Eastern Europe can get an EFFO membership free of charge.

Various Items from the Agenda

Plans for 1995 include the preparation and publication of 3 issues of OS-9 International as the main item. It is considered vital for OS-9 International that authors other than the publishers themselves submit articles. But it is still unclear how to reach and to convince them. At least, we take the opportunity of these minutes to repeat our call for papers. Apart from these publishing activities, enhancement of and additions to our PD disk collection are planned. This includes:

- Remote commands for TCP/IP networking. The software is already available and currently being tested.
- TEX, which is an open item for quite a long time is planned to be made available together with an issue of OS-9 International dedicated to text processing.
- The current version of μ-emacs (me 3.12) has been ported to OS-9 by Hubert Nehring and will be available soon.
- *gmake*, the GNU make is now also available and running very promising. It is expected that this PD will receive maximum attention as *gmake* provides important enhancements compared to what is currently used under OS-9.

Please note that these disks may not be ordered unless they appear on our list of officially released PD disks.

The OS-9 mailbox available via the dial-up network at the Federal Institute of Technology in Zurich (KOMETH) has been shut down last December, because the host system has been switched off. This did not cause major problems since it was only rarely used in the past. The mailbox functionality will be replaced by distributed mail servers; more details will be given in the next issue of OS-9 International.

EFFO's Internet name domain *effo.ch* proved to be very useful. The contact to OS-9 users worldwide via Internet is as important as the EFFO fax or conventional mail. The committee members can be reached via the following addresses:

Hans-Werner Bippus	hwb@effo.ch
Carsten Emde	carsten@effo.ch
Stephan Paschedag	stp@effo.ch
Reto Peter	reto@effo.ch
Werner Stehling	stehling@effo.ch

In addition, two mail addresses have been installed that are used for general questions about EFFO or OS-9 International:

EFFO	effo@effo.ch
OS-9 International	os9int@effo.ch

_getsys();

Reto Peter

Monthly EFFO Meetings

The meeting always takes place every first Friday of a month. We meet in Brugg, Hotel Rotes Haus, either in the restaurant (at 7 PM) or in our meeting room in the basement (at 8 PM).

Everybody interested in OS-9 is kindly invited to join the meeting.

Publishing Dates of OS-9 International

Three issues of OS-9 International will be published per year. The shipping dates are beginning of February, June and October. Editorial deadlines are December 31, April 30 and August 31, respectively. For advertising conditions please contact EFFO.

Imprint

OS-9 International

Published by	European Forum For OS-9 (EFFO)
President	Werner Stehling
Vice President	Reto Peter
Director of Finance	Stephan Paschedag
Editor-in-Chief	Carsten Emde
Design	Marc Balmer, Werner Stehling (layout)

Address
European Forum For OS-9
P.O. Box
8606 Greifensee
Switzerland

FAX +41 1 940 38 90
email os9int@effo.ch

Printed directly from disk by Fotoplast, Zurich, Switzerland
ISSN: 1019-6714

Subscriptions

OS-9 International is the official organ of the European Forum For OS-9 (**EFFO**). The subscription is included with the annual **EFFO** membership fee. In addition, it is available by separate subscription for non-**EFFO** members, single issues are also available. All following prices are given in Swiss Francs, shipping included:

	Switzerland	*Europe*	*Overseas*
One year (3 issues)	25.00	30.00	35.00
Single issue	10.00	12.00	14.00

To subscribe to **OS-9 International** or to order a single issue send a letter, postcard, fax or email to **EFFO**.

Advertisements

OS-9 International is not only an ideal platform for discussing OS-9 related topics, it is also the ideal place to advertise. **OS-9 International** reaches end-users, system-software developers and, nevertheless, decision-makers.
Please contact **EFFO** for detailed information on how to place an ad in **OS-9 International**.

EFFO *European Forum For OS-9*
8606 Greifensee
Switzerland
Fax +41 1 940 38 90

What is EFFO and who should be interested in it?

EFFO stands for *European Forum For OS-9* and was founded in 1988; its main goal is to support Microware's multi-user multi-tasking operating system OS-9 that runs on the 68k family of Motorola microprocessors. This support primarily consists in providing a means of communication between people who already use and appreciate OS-9 and those who do not yet but would profit by doing so.

EFFO is independent from and not commercially related to any company. Its members are companies offering OS-9 compatible hardware, OS-9 system programmers, computer clubs as well as end users such as private computer owners, research institutes and university departments.

EFFO intends

- to provide a collection of public domain software that is of general interest and that helps to make OS-9 more attractive to programmers and users,
- to coordinate ports of (mainly UNIX) software (e.g. to avoid that a particular software is ported many times and other equally important software still is not),
- to make all the nuts and bolts of managing an operating system available to everybody so that "the wheel has not to be re-invented all the time".

Until the end of 1992, the rules of the game were that disks containing the EFFO software pool were available without charge, and everybody taking part in this service was expected to make contributions to the pool. Initially, this concept – although somewhat idealistic – worked quite well. Over the years, however, less and less contributions have been made so that, starting in 1993, a new concept was created.

First the bad news: The distribution is no longer free of charge, there is a handling fee in an order of magnitude comparable to all other PD pools still being incompatible to OS-9.

And now the good news:

- The disks contain ready-to-use software that has been thoroughly tested.
- The software is maintained and updated continuously.
- All disks come with printed guidelines of how to install and to use the software. Some of them even have a complete user's manual in printed form.
- The 23 forum editions and the 10 PD disks representing the EFFO software pool as at end 1992 will – although no longer maintained and upgraded – still be available.

In addition, EFFO has a printed forum: the journal *OS-9 International*, published by EFFO, is devoted to OS-9 related topics. Every issue includes the most recent version of EFFOs public domain software list. This list will also be made available on the EFFO bulletin board and through regular postings to international network boards.

Please contact EFFO at

EFFO
P.O. Box
CH-8606 Greifensee
Switzerland
Fax +41 1 940 38 90
email: effo@effo.ch

to obtain more information. This is also the address where the editorial staff of *OS-9 International* and all active EFFO members can be reached in case of questions concerning particular articles or software packages.

Last but not least we invite you to join EFFO. As a regular member you get some price reduction on our PD disks, and a one-year subscription to *OS-9 International* is included. Your membership will be very helpful not only in financial aspects but also as moral support: a bigger EFFO can move bigger mountains, can't it? For enlistment of three new EFFO members you get a free membership for the next year.

Notes concerning the PD collection

- The updated list is regularly published in *OS-9 International*.
- Version numbers normally reflect the number assigned by the author. Collections of several distinct items get a version number assigned by EFFO. Usually we deliver the newest versions available.
- The disks are 3.5" in universal format. Normally OS-9 version 2.4 or higher will be required.
- Normally the program is free for personal use. In general, the GNU licence or the copyright notice of the author must be respected.
- Prices are calculated on a per disk base. The units are Swiss Francs or Deutschmark (to keep things simple we use a 1:1 exchange rate).
- The following issues of *OS-9 International* are still available:

❑ 1/92 ❑ 1/93 ❑ 2/93 ❑ 1/94 ❑ 2/94
❑ 3/94

CB, WS 95/1

EFFO OS9 **European Forum For OS-9**
8606 Greifensee
Switzerland
Fax +41 1 940 38 90

EFFO PD Collection — Order Form

valid as from **3. Feb 95**

Nr.	Title	Vers.	Contents	Status	# of Disks	Order
100	Utilities	1.1	*diff* compare files; *upatch* inverse to diff; *fgrep; mkdir* builds complete path; *move; space, dinfo, rendisk: disk utilities; help:* like VAX/VMS; *cheksum, fillup, rawcopy, exb:* EPROM utilities		1	
101	SC	6.21	*sc* spreadsheet calculator		1	
102	GCC	1.42.0	GNU C compiler (in compliance with ANSI), executables only; about 2 MB RAM needed		1	
103	GCC sources	1.42.0	GNU C compiler, sources		3	
104	GPP	1.40.3	GNU C++ compiler, executables only		2	
106	Ghostscript	3.12	Postscript level 2 interpreter for output devices such as various HP printers, other printers, MGR window manager, GIF/PCX/PBM/PGM/ PPM file formats, etc.		4	
107	Shell	1.7	*sh* combined and enhanced features of the Microware and the Bourne shell, full VT100 support	update version #44	1	
108	Communication	1.1	C–*Kermit* 5A(188); *inout* connects two SCF devices; *Z–Modem* 3.17		1	
109	Network	1.0	*SLIP* (ex KA9Q): TCP/IP via RS232		1	
110	Network support	1.0	remote printer *lpr*, remote shell *rsh*, remote copy *rcp* and remote time services *netime, remotime*	new	1	
112	OS–9 Lib	90/3/18	set of procedures available on UNIX-systems to allow and ease implementation of UNIX-software on OS–9 systems		1	
113	Communication source	1.1	*C–Kermit* source 5A(188), *Z–Modem* source 3.17, executables are on PD#108		2	
117	GCC	2.5.8	GNU C compiler (in compliance with ANSI), executables only; about 4 MB RAM needed		2	
119	GPP LIBGPP	2.5.8 2.5.3	GNU G++ compiler, LIBG++, executables only; about 4 MB RAM needed		4	
121	OS-9 International	1.0	programs that appeared in OS-9 International and accompanying programs; covers up to and including issue 3/94		1	
123	OS9exec	1.14	OS-9 cross development on the Macintosh, single task, no realtime. **Note:** this is a Macintosh disk		1	
				total number of disks		

				Unit Price	Qty	Price
PDs	Software	1995	total number of PD disks (normal / EFFO member price)	15.– / 10.–		
DOC	Docu box	1992	hardcover folder with box to collect manuals (A5 size)	30.–		
HDL	handling	1992	package and handling (add with disk or box orders only)	10.–	1	
ESM	single member	1995	EFFO single membership (private users)	80.–		
EGM	group member	1995	EFFO group membership (companies, institutions, computer clubs, users groups)	150.–		
SUB	subscription	1994	separate subscription to 3 issues of *OS–9 International* (included with ESM and EGM). The three prices are for shipping to Switzerland, Europe or Overseas, respectively.	25.– 30.– 35.–		
ISS	single issue of OS-9 International	1994	❑ 1/92 ❑ 1/93 ❑ 2/93 ❑ 1/94 ❑ 2/94 ❑ 3/94 The three prices are for shipping to Switzerland, Europe or Overseas, respectively.	10.– 12.– 14.–		
				Total Amount (SFr / DM)		

Name .....

Address .....

City .....

Country .....

Phone .....

Fax .....

Date / Sign .....

Sorry we don't accept credit cards or cash. Add 5% for checks. Please prepay to one of the following accounts:

Switzerland:
Postal Giro Account, 80–48 254–4 (Zürich)

Germany:
Volksbank Jestetten eG, 111 2007 (BLZ 684 915 00)

❑ Please bill me in advance

❑ I am already EFFO member

my name and address is

❑ confidential ❑ for members only ❑ free